Using extended web technologies to develop Bluetooth multi-platform mobile applications for interact with smart things

Jordán Pascual Espada a, Vicente García Díaz a, Rubén González Crespo b, Oscar Sanjuán Martínez c, B.Cristina Pelayo G-Bustelo a, Juan Manuel Cueva Lovelle a

a Department of Computer Science, University of Oviedo, c/Calvo Sotelo, s/n, 33007 Oviedo, Asturias, Spain
b Department of Computer Science, Pontifical University of Salamanca, Madrid, Spain
c Department of Computer Science, Carlos III University, Madrid, Spain

Article history:
Available online 6 May 2013

Keywords:
Mobile Web Applications
Web information fusion
Context-aware communication
Bluetooth
Web browser
Internet of things

A B S T R A C T

Nowadays the classic web paradigms are being subjected to changes; every day millions of users around the world use their Smartphones to access web applications from anywhere. The World Wide Web is one of the biggest repositories of information in the world, and that information is stored in internet servers and repositories, but today in the real world there are many other information sources such as electronic devices with communication capabilities: smart appliances and sensor networks. The Smartphones are equipped with communication hardware elements like the Bluetooth module, which allows the Smartphone to exchange information with nearby electronic devices. Every day more and more mobile applications are being developed for native platforms that use Bluetooth’s communication module to send and receive information from different sources. Native mobile applications use the specific platform’s APIs to manage the Bluetooth communication actions (send and receive information, search for devices, etc.), however, web applications do not have technical capabilities to manage the Smartphone’s Bluetooth communication module and thereof cannot use that kind of information. The main objective of this research work is to design a novel framework that allows classic web applications to use information from nearby electronic devices. The proposed framework must be easy to use and able to be integrated with common web technologies. Developers can use this framework to include new information sources and data exchange procedures in an easy way. The new type of information can be merged with the web to develop or improve algorithms and web applications.

1. Introduction

Information fusion refers to the study of techniques that combine and merge information and data residing at disparate sources [1–3]. The information fusion is present in many technological areas, including the World Wide Web, where different types of information fusion play an important role. The web influences almost every aspect of businesses and our daily lives [4].

In many senses, Smartphones have become a revolution within the last years. The increase of the computing and communication capacity of these devices allows them to perform a wide range of tasks. Millions of users use their Smartphones daily to perform lots of tasks, such as internet access, global positioning systems, multimedia, smart homes control [5], remote control [6], and patient monitoring [7]. Among the current mobile platforms we must highlight Android, iOS, Windows Phone and Blackberry, which are the target of millions of mobile applications that are distributed via app stores. Many of these mobile applications merge information from the internet with other information obtained by using the phone’s hardware elements, such as Bluetooth, Wi-Fi, sensors, and GPS.

The power of processors and the hardware communication elements of the Smartphones have made possible for many mobile applications to base a large part of its functionality in the exchange of information with other nearby electronic devices, thus enabling the development of context aware applications. Within the field of wireless communication technologies, the Bluetooth is one of the most used. Currently, most of the Smartphones include this technology. This communication standard is used to communicate devices both at industry [8] and at home [9], and it is also common to use it in multiple remotely controlled systems for fields like robotics [10] and telemedicine [11].

The most popular mobile application markets contain many applications that exchange information with nearby electronic devices, such as Smart Remote Control, LG Bluetooth Remote,
Samsung Remote, DSLR Remote Controller, Logitech Squeezebox Controller and NXT Remote Controller, among others. This type of data exchange functionalities becomes very useful nowadays and the acceptance of many of these applications is reflected in the number of users, ratings and reviews provided. Samsung Remote Android application has gained more than 750,000 users in the second half of the year 2011. Currently, these mobile applications often exchange information with sensors, networks, home appliances (Smart TVs, Smart fridge, monitor electricity consumption, etc.), Bluetooth accessories, health monitoring devices, etc. But the number of physical objects that include communication mechanisms is growing rapidly; these objects are commonly called smart devices or smart things. These smart devices are one of the key pieces from the internet of things (IoTs) paradigm that promotes systems based on the communication between smart devices [12].

From the development point of view the management of the Smartphone's communication modules is done by using a platform's own API. Currently, the most popular Mobile Platforms are: Android, iOS (iPhone), BlackBerry OS, Symbian OS (Nokia), and Windows Phone 7 & 8 and Bada (Samsung), therefore, native applications are the only ones that usually include this type of information exchange. The mobile market is currently divided in multiple mobile platforms. In many cases, the notable differences between platforms (programming languages, APIs, interfaces, etc.) mean that in most of the cases, if the developers want to create a mobile application with repercussion on a large number of users, they should develop the same application multiple times, following the procedures for each mobile platform.

If we intend to make possible for a great number of users to exchange information with the smart electrical appliances of their kitchen via Bluetooth it is probable that we have to develop a native mobile application for every platform, adapting to each user's mobile platform. These applications would have a very different implementation in lots of its modules, as the management of the communications via Bluetooth will depend on the platform (specific classes, packages, methods, listener, authorizations, etc.). The mobile platform sector's current heterogeneity implies that, in order to offer mobile applications with advanced characteristics (such as the access to hardware communication modules) and make possible for the user community to use them, it would be necessary to multiply the development processes. This involves a very expensive process, as it requires to re-implement the same application several times and forces the developers to have knowledge of the various peculiarities of each platform.

The Smartphones users cannot only use native mobile applications, but also web applications. Implementing and maintaining the same application on multiple mobile platforms involves high development costs, so far that sometimes many mobile developers choose to develop their applications as web applications. With this strategy they are not only able to reduce development costs but they also increase the number of potential users, because the application can be used on any mobile device that has a web browser, regardless of the mobile platform [13]. But the nature of web applications makes impossible for them to manage the mobile’s communication hardware, so the data exchange process with other electronic devices using Bluetooth cannot be performed. Native mobile applications can use the platform API, while web applications cannot. These APIs can communicate directly with the device's hardware elements, including: sensors, cameras, Wi-Fi and Bluetooth communication elements [14]. These hardware elements can allow the capture and exchange of many different types of information; it is very different from the classic information that is commonly used on the web.

There are other approaches based on the use of distributed mobile applications, like PhoneGap [15,16]. This type of distributed applications could be considered hybrids of native and web applications in many ways, because they use web technologies and a set of specific libraries to develop an application for a specific mobile platform. The specific libraries provide many of the mobile communication features that the web technologies lack, such as access to the Smartphone's Bluetooth module to send and receive information. But in any case, applications built with these hybrid platforms can be considered as hybrid web applications. They simply extend web technologies to make them useful for developing mobile applications. These types of platforms do not aim for the current web applications to exploit the hardware features of Smartphones, like Bluetooth, for getting information from new sources.

The main objective of this research work is to design and develop a framework that allows classic web applications to integrate information from nearby electronic devices, such as smart appliances and sensor networks. With this novel approach, the classic web applications will be able to use information from new sources, allowing them to implement new features depending on the information from the user’s physical environment, such as nearby electronic devices. This framework must be easy to use and should be able to be integrated with existing web technologies. The aim is to make possible for developers to use this framework in order to include new information sources and new data exchange procedures in an easy way. The new information will be used to develop or complete algorithms and applications on the web. This approach gives developers the possibility of merging the web information with information from different nearby electronic devices.

This proposal is based on the use of extended web mobile applications which contain a series of specific XML tags that make possible to model all the Bluetooth communication’s actions (search for devices, send information, receive information, etc.) in a fast and particular way. This tags contained in the web applications are combined with the use of traditional web technologies and are interpreted by a particular module of the web browser called “communication actions module” that accesses the Bluetooth communication module from the client mobile device, enabling the exchange of information with nearby electronic devices.

2. Related works

The use of mobile devices to control some functions of other electronic devices comes from a long time ago, in 1997 at Carnegie Mellon University’s the Human–Computer Interaction Institute (HCII) launched the Pebbles project one of the pioneers systems based on the use of PDAs as effective remote controls [17].

In these last years, the number of electronic devices that are equipped with the necessary hardware to send and receive information from other devices (such as mobile phones) has increased greatly (home and office appliances) [18]. The technological evolution of the involved devices has led to several different kinds of information exchange and control systems. There are many factors that can determine the system's structure, such as hardware and software components, the system objectives, device specifications, communication types, and environment features.

Home networks are the target of many commercial and research projects that use mobile devices to exchange information and remotely control other nearby electronic devices. Usually, these systems have some limitations such as heterogeneity of controlled devices that use different protocols: X-10, Bluetooth, Universal Plug and Play (UPnP) [19]. Among the different kinds of control systems we can identify, there are some that use web servers and services to provide a centralized control environment capable of managing multiple different devices. Most of these web systems focus on network appliances [20]. Many of these systems
provide web-based control interfaces that can be used by different platforms devices [21], but this kind of system does not exploit the communication hardware elements of the client mobile device (Bluetooth, Wi-Fi, etc.), the web server uses its own communication mechanisms to establish the communication with nearby electronic devices. In this system the client mobile device is limited to communicating with the local web server. Therefore, the environment requires placing a local web server near the electronic devices that will be controlled remotely.

Other types of data exchange systems are based on simpler architectures that do not require specific hardware components. Generally, these systems use a software component that runs on mobile devices. These software components are often the most affected by the diversity of mobile platforms. Most of the commercial remote controlled applications are platform dependent, such as Samsung Remote for Android. Other applications are device dependent, such as Philips Pronto applications.

Several research projects introduce multi-platform software components that can be used to build remote control systems [22]. In this type of systems the remote control software component works as an adapter. In this case, the adapter software is responsible for the connection with devices and also the information that is being sent and received. The software component can be transferred and executed by different mobile devices. Each software component has the capacity to remotely control one or more devices. That way, mobile devices can run multiple software components that allow them to remotely exchange information and control many electronic devices, including electronic devices with different communication protocols and specifications [23]. The approach presented for systems based on software components as adapters is suitable for many scenarios. Adapter software components have to be downloaded in mobile phones; they also contain executable code that is potentially dangerous. The downloading process of the components adapters can be very slow the first time the user configures the remote control. That's why this approach is not appropriate for scenarios where people occasionally interact with devices.

Several proposals have provided some specific native mobile applications capabilities to web applications. This kind of proposals allows mobile web applications to use contextual information from the hardware elements of the client mobile device, such as sensors or GPS [24,25]. But none of these proposals provides mechanisms for web applications that can manage the client’s device hardware elements that provide communication capabilities, such as Bluetooth. Such is the impact and potential utility of this type of communications that System Applications Working Group - W3C considered to include a standard API in the future to implement Bluetooth communications in the web, within the specification of HTML5. The first specifications of this functionality are included in the second phase of the Roadmap, which starts when the first one ends. It is scheduled for Q2 2014 [26].

At the moment, the only Bluetooth communication API specifications related with the web are in hybrid platforms as PhoneGap [15] or Tizen [27]. These platforms which use web technologies to facilitate the development of hybrid mobile applications [28], which run locally on the Smartphone in a similar way to native applications, but are not aimed to integrate Bluetooth communication on the web. These platforms use JavaScript libraries to specify Bluetooth communication processes; these libraries have a very similar approach to that of the native Bluetooth APIs. This research will provide a solution to include Bluetooth communication capabilities in the web, which will allow the fusion of web information and information from nearby electronic devices in an easy and dynamic way, specially designed for the web.

3. BT communicator web browser

3.1. Considerations and design decisions

Nowadays, web applications do not have the capacity of managing hardware elements such as the client device’s Bluetooth communication module. The main purpose of this solution is to allow web applications to manage that module (Fig. 1), allowing a detailed specification of the communication process in an easy way and regardless of the platform, focusing only on the communication actions such a connections, pairings, the sending and reception of data.

This solution is based on client–server architecture. The client part uses a web browser with an extended functionality called “BT Communicator web browser”, which is constructed according to the proposed specification. This web browser can detect requirements that specify Bluetooth based communication processes and access the Bluetooth communication hardware elements of the client mobile device. The actions performed on the client device’s Bluetooth communication module will depend on the specification of the communication contained in the web application, such as connections, pairings, and the sending and reception of data.

The specification of the communication processes will be performed by a set of “BT communication action XML tags” that will be included in the web page’s presentation layers.

One of the main objectives of this research is to reach a new model for the development of multiplatform mobile applications with a functionality based on the use of the device’s Bluetooth communication module. Specifically, applications that can communicate among themselves or that can remotely control other nearby smart things or electronic devices. To reach this objective we will develop both a specification and an architecture for the “BT Communicator web browser” independent from the platform, allowing the development of an implementation of the browser for most of the current mobile platforms (Android, iOS, Blackberry OS, Samsung Remote for Android. Other applications are device dependent, such as Philips Pronto applications.

Fig. 1. Conceptual scheme. A distributed application on the web is an application that can be used from any mobile platform. The BT communicator web browser allows the Smartphone to communicate with nearby devices using Bluetooth. To execute that communication the BT communicator web browser processes the communication actions described in the web application. BT communication actions are responsible for the exchange of information with nearby devices.
Symbian OS, Windows Phone 7 & 8, etc.). But even if the architecture and the specification of the functionality is independent from the platform, it is unavoidable for some parts of the architecture to have snippets of code that are platform dependent, specifically those parts of code that access the operating system or specific libraries that are platform dependent, because each platform has its own APIs and programming languages.

One of the most common negative aspects in the development of mobile applications that require Bluetooth technology is the difficulty of implementation, which requires using specific classes and methods and the developer usually has to learn many procedures to work with it. Depending on the mobile platform, the specification of Bluetooth communication completely changes and can imply the use of several specific libraries, methods, permissions, etc. The approach in this research also aims to drastically simplify the specification of the Bluetooth communications, allowing the developer to focus in the specification of the communication actions instead of in the implementations that are platform dependent. With this approach, the specification of communications becomes a light and fast process from the implementation point of view and in comparison with the development of native applications in other platforms.

To avoid limitations in any aspect of the development of web applications, the specification of a Bluetooth communication process should not be linked to a particular web technology, and should not require use libraries or frameworks that make the development of web application more complex or dependent on other technologies. The web application will only use a set of special XML tags “BT Communicator web browser”. These XML tags are inserted into web pages and they are typically interspersed with the HTML or JavaScript code.

The information that is obtained through the communication processes and has to be sent to the web server (the list of nearby devices, a response, the reception of data, etc.) will be transmitted using http POST petitions that can be processed by web pages or standard web services. This services can be implemented by using any programming language that is commonly used for the development of web applications (PHP, Java, C#, Python, Ruby, etc.).

3.2. Model

The proposed system consists of two parts:

1. In the web application we use a specific language consisting of a group of BT communication action XML tags that define the required Bluetooth communication actions. The tags that specify the communication process are inserted into the presentation code of web pages, commonly in the HTML or JavaScript code.

2. The BT Communicator web browser performs the analysis and post processing of the communication action XML tags. This layer translates the XML tags into actions to be executed by the device’s hardware, specifically the Bluetooth module. It is also responsible for sending the target device’s responses to the web application (Fig. 2).

3.3. Bluetooth communication specifications in Mobile Web Applications

The Bluetooth’s communication action XML tag <Bluetooth>, which is associated with the Smartphone’s Bluetooth communication module, defines a group of XML attributes that contain the action’s name (scan, connect, pair, send, etc.) and other relevant information for the communication process, such as the device’s name, and the message that has to be sent. This attribute group defines the communication process for Bluetooth technology in a simple and highly detailed way. The markup language supports most of the common communication actions, so it will be considerably more restrictive than the system Bluetooth APIs used by native applications.

Some of the specific attributes that are used to specify the Bluetooth communication processes are noted in Table 1. Attributes must be combined depending on the actions that we want to be performed. For example, the “Data” attribute must be used only when a “Write” action is being performed.

Another group of attributes specifies the graphic representation and the process of user interaction with the web application, such as the redirection of the URL after completing the communication action or the URL to which the device’s response to the communication action is sent. The attributes are shown in Table 2.
These XML tags are included in the presentation layer of the web application, usually in the HTML code. BT communication action XML tags can be dynamically generated using the most popular dynamic languages for web application development: PHP, JavaScript, Java, Python, C#, etc.

Below are some examples of communication action XML tags:

```xml
<!-- Scan Bluetooth devices and send the result to the Web Service scan.php -->
<bt-action action="scan"
receiver="scan.php"
x="50" y="45" image="btn.png"/>

<!-- Pair Bluetooth with the device "MyCenter", the device has been previously scanned -->
<bt-action action="pairDevice"
deviceName="MyCenter"
receiver="scan.php"
x="50" y="45" image="btn.png"/>

<!-- Connect with UUID service in a device, search the name of the device in real time (scan for devices) -->
<bt-action action="connectDevice"
UUID="5ee01872-4c9e-48d6-9f61-015ff816b278"
x="50" y="45" image="btn.png"/>

<!-- Send data to a connected device, requires an active Bluetooth connection -->
<bt-action action="write" data="1"
x="20" y="55" image="btn.png"/>

<!-- Read data from a connected device forwards the information to the web service, requires an active Bluetooth connection -->
<bt-action action="listen"
receiver="listenData.php"
x="20" y="55" image="btn.png"/>
```

The dependence between the performed Bluetooth actions is important when defining a communication process. The Bluetooth actions alone have no value, so the execution order and the result of the previous executions is important. Data cannot be sent if it has not been paired and connected to a device.

The Bluetooth actions will be executed only when the user of the client mobile device gives its permission, that’s why it is advisable to display the Bluetooth tag acceptation button in the specific order to correctly perform the communication process.

The next scheme (Fig. 3) shows the dependence order that exists between the specified Bluetooth actions:

3.4. BT communicator web browser

The main functionalities of the Communicator web browser are divided into three sections:

- Classic interpretation of web pages: this module processes the standard web languages (HTML, CSS, JavaScript, etc.) and other common technologies that are used in popular application development. This part of the functionality is equivalent to commercial web browsers.
- Inform the user about the communication actions contained on the web: When a web application contains Bluetooth communication action XML tags, the browser has to notify the user that is a request for some kind of communication. In order to inform the user, the browser displays a special button in the web interface. This button shows the current status of the communication action, there are four possible states: (1) ready, (2) processing, (3) completed and (4) error.
- To manage Bluetooth communications between the web application and the target device, depending on the processed communication action XML tags.

The architecture of the communication manager web browser is composed of three layers:

3.4.1. Layer 1: The standard web browser

Its aim is to analyze and interpret the web page code that forms web standards and provides a graphical representation of the page. This functionality is equivalent to the one offered by commercial web browsers (Chrome, Firefox, Opera, etc.).

3.4.2. Layer 2: The Bluetooth communication action XML tags manager

Its objective is to manage of Bluetooth communication action XML tags. In the first stage of the process, the BT communicator browser analyzes the page code looking for elements that meet the definition of the communication action XML tags. This XML tags represent a communication requirement that must be notified to the user, the browser will render each communication action XML tag on the web page. The graphical presentation of these communication requirements will be specified by buttons on the browser's window. The text button is defined in the ‘image’ ‘x’ and ‘y’ attributes. Also, the browser includes a status bar that notifies the user about the current status of the communication action: Ready – Processing – Complete – Error.

![Diagram](image.png)

**Fig. 3.** Scheme dependence between Bluetooth actions.
3.4.3. Layer 3: Communication manager

This layer is responsible for the management of communications between the web application and the target device. The communication manager is the most complex part of the BT communicator web browser.

When the user press the button that initiates an action, the Bluetooth communication action XML tag is processed by the communication manager module. First, the module runs a series of regular and specific checks, because communication actions require other actions to be previously and successfully executed. For example, it must be connected to a target device before sending the data. To keep Bluetooth connections alive and monitor their status, the communicator manager module uses a dynamic storage of communication processes. This communication processes monitor the connection threads and record the communication action that runs on each one.

One of the most important parts of the communication manager is the transformation module; this module applies a series of algorithms to transform the XML tags and its attributes in one or more Bluetooth API system calls. Although the objectives of the communication actions XML tags are the same on all platforms, the transformation system depends largely on the platform, because each one uses its own API.

These API system calls access to the hardware communication elements of the device. The communication manager module saves the communication status after the execution of each action, so the combination of independent actions can generate a complex communication process. Sometimes, the target device responds to communication actions or starts sending information by using an open communication process. It uses an HTTP request POST type to send information from the device to the web server, using the URL specified in the ‘receiver’ attribute of the communication action XML tag. The request that is sent to the server requires additional information that is contained in the parameter Device Session Identifier.

- Device Session Identifier (DSK): This parameter is used to uniquely identify the navigation session and the specific device that made the request. This session ID provides a similar functionality to other common web technologies (PHP, JavaScript, C#, Python, etc.), but it is also generated depending on the identity of the client device. The DSK is contained in all the requests received from the communication manager web browser.

By using these http headers the web application can check the source of the received request.

4. Prototype: Multimedia remote controller

Our goal with the development of this prototype is to demonstrate that the proposal is capable of merging information from the web and a nearby electronic device. To illustrate this feature, we have selected an information exchange real scenario. We are going to develop a web application that allows users to control a Multimedia Center using a mobile Bluetooth application, similar to many native Android and iPhone applications that allowing remote control an electronic device. This section details the development process for a web application that uses BT communication action XML tags. The proposed solution will be used to develop a web application that is capable of exchanging information with nearby electronic devices. With this proposal, manufacturers of electronic devices can deploy multi-platform web applications that are technically capable of interacting with smart things and electronic devices. Also, by using web applications many people (family members, office personal, hotel guests, etc.) can interact with the electronic devices in the environment in a simple and fast way.

The developed web application uses the BT communication action XML tag’s specification to model a communication process based on Bluetooth technology. Several typical actions are used in this process, like the establishment of connections and the exchange of data between devices. The developed web application will be interpreted by an Android version of the BT communicator’s web browser. One of the main purposes of this prototype is to illustrate the development process for web applications that use BT communication action XML tags in order to implement new communication possibilities with nearby electrical devices. The selected functionality for the web application is similar to the one used for a remotely controlled multimedia center. Normally, these kind of remote controllers are implemented as native applications, because they need to manage the communication mechanisms of the Smartphone for an exchange of information between the two devices.

The web application has a single index.html page, from which the user manages the connection process and the sending of commands to the device (Computer) that is executing the Bluetooth service. The service implements the logic that allows to remotely controlling the multimedia center. To define the way in which the web application must establish connection with the multimedia center device the tag <bluetooth/> is used in combination with the action = “connectDevice” attribute. This attribute indicates that a new connection must be established.

When the communication XML tag defines a “connectDevice” action it can be accompanied by other attributes, like: “device-Name”, to define the name or direction of the target device, “UUID” to define the service’s identifier or “pair”, which is used to indicate if the connection requires a previous pairing between devices. In this case, the target device can change depending on the target computer for the communication, so the “deviceName” attribute shouldn’t be used. Due to this attribute’s omission, the web browser executes a scanning of nearby Bluetooth devices and allows to choose one of them as the target of the connection. In this case, the “UUID” attribute will be necessary, as it allows to identify the service we have to connect to, in other words, the service that contains the business logic for the reception of the commands that are applied on the multimedia player. The service’s UUID will always be the same, regardless of the device that is executing the service. The device pairing that is specified with the “pair” tag has a true value by default, so “pair” can be omitted if it is intended to do a pairing before the connection. If it is necessary to specify that a previous pairing is not needed it would be necessary to check that the target device is ready to engage insecure connections.

Aside from the attributes that configure the communication action, there other attributes that will be added to the <bluetooth/> tag. First of all, the “x” and “y” attributes will be added to specify the position of the graphic representation corresponding to the context XML tag. The final syntax will be like this:

```xml
<bluetooth action="connectDevice"
    UUID="5ee01872-4c9e-48d6-9f61-015ff816b278"
    x="50" y="45"/>
```

Once a connection has been established between the devices, the web application must allow the user to select the commands that are being sent to the multimedia center, with the purpose of remotely control its running. The web applications will allow to execute six different actions:

1. Rewind the reproduction.
2. Start the reproduction.
Each of these actions is defined using a numerical code, which will be interpreted by the service to execute the proper functionality over the multimedia player.

Each number corresponds to a sending of information, to define these processes the `<bluetooth>` XML tag will be used along with the `action = "write"` attribute. This kind of action requires the use of the `data = "bytes"` attribute. By using these attributes, we can specify the message that will be written on the connection thread and, therefore, the data that is going to be received by the target device once the action is executed. For the data sending communication actions to be successful, the web browser must be connected to the multimedia center (Fig. 4). The tasks associated to the tags that define the sending of the six commands will take the previous established connections as a reference. When applying several communication action XML tags of the same type it is almost essential to use specific attributes that allow the customization of the graphic representation.

This kind of web application is particularly important in order to reach a graphic aspect that favors the user’s experience. That’s why the “image” attribute will be added to the six tags that are responsible for the transmission of commands.

The index.html page’s code contains the communication action’s XML tags which define the information that will be sent to the target device is shown below:

```
<bluetooth action="write" data="1"
  x="20" y="55" image="backward.png"/>
<bluetooth action="write" data="2"
  x="40" y="55" image="play.png"/>
<bluetooth action="write" data="3"
  x="60" y="55" image="stop.png"/>
<bluetooth action="write" data="4"
  x="80" y="55" image="forward.png"/>
<bluetooth action="write" data="5"
  x="40" y="65" image="up_vol.png"/>
<bluetooth action="write" data="6"
  x="50" y="65" image="down_vol.png"/>
```

This web application is not essential to include a `<bluetooth action="disconnectDevice" ..."/>` tag, as the connection between the two devices will be automatically cut when the user exits the web application. The BT Communicator Browser’s screen shows specific buttons that are associated with the communication actions described in the web page. The first tag defines the connection process and uses the Bluetooth task’s default image. The rest of tags use custom icons.
When the user clicks on the button associated to the `<blueooth action = "connectDevice" />` XML tag, the corresponding programmed context task begins. A target device has not been specified, so the first step on this task consists on performing a scanning in order to find nearby Bluetooth devices (Fig. 5).

To continue with the connection task, the user must choose from the list of found devices, specifically the one which executes the service that allows to control the multimedia center. There is no specific reference to the paired in the BT communication action XML tag that defines the connection process, so the connection will require a previous pairing of the devices (Fig. 6).

After the pairing, the connection process goes on for a few seconds. A change in the visual aspect of the button associated to the communication action XML tag tells the user that the connection has been successfully established. From this moment, the sendings defined on the other six BT communication action XML tags will take the established connection as a base. When the user clicks on one of the buttons associated to the XML tags, the web browser executes a sending of data towards the service with the specified UUID. The transmitted data contains the identifier code of a command. These commands unleash the execution of different events that allow the management of the multimedia player. The target device executes a service that has been implemented by using java.

Initially, the service waits for the connections to pop up, and when this occurs the command reception process begins. Commands are chains of bytes. These chains are analyzed by the service’s business logic in search of coincidences with the command identifiers that have been programmed. When a match is detected, an action is executed on the multimedia center, like changing the volume (Fig. 7).

5. Proposal evaluation

Given that the developed prototype has successfully executed a distributed web application with a great part of its functionality supported on Bluetooth communications, we can conclude that we have managed to design a system that is platform independent. The only requirement is that the Smartphone must have a web browser developed under the BT communicator web browser specification, and from this point the same application will work in the same way in different mobile platforms.

Another objective of this approach was to simplify the implementation of the applications that used Bluetooth as communication mechanisms and abstract the mobile platform’s elements from the specification of a communication process based on communication actions.

Once proven that the functionality meets the objectives, we must analyze the implementation of the application and see if it has an acceptable degree of simplicity compared with other developing technologies. For this evaluation, we will analyze the implementation of several applications with a similar functionality on different platforms.

The analyzed applications present a basic use of the Bluetooth communication module, the applications are paired with another device and exchange the information with it. For this evaluation we have analyzed the developed applications with some of today’s most influential mobile development technologies:

- Java – Android
- Javascript – PhoneGap

PhoneGap is the most popular mobile cross-platform, and is the only one that supports Bluetooth.
- Objective-C iOS
- PHP + BT communication action XML tags.
The evaluated applications have been developed with a minimum implementation, which makes possible for them to perform the entrusted task.

The measured aspects in the source code and the application’s structure are the ones below:

- \( L \) = Number of lines of code added to the application in order to give it the specific functionality required for every case.
- \( T \) = Size: number of characters, spaces included.
- \( E \) = Number of specific elements (classes, tags, objects, etc.) that are required to manage that hardware element.
- \( F \) = Number of additional configuration functions that are required aside from the implementation, such as permission assignment and associate external libraries.

The results of the analysis are shown in Figs. 8–10.

With the purpose of establishing a global comparison mechanism that can estimate the complexity that the development implies, a weight has been assigned to each of the measured aspects \( L = 2, T = 1, E = 3, F = 4 \). The Fig. 11 shows the global results of the measured aspects, according to their weight.

Taking into consideration the analyzed aspects and the criteria of this analysis, we can conclude that the specification of Bluetooth communication can be easier with this approach than with the ones that are commonly used in other native platforms (Android – Java or Objective-C iOS). It also has a much more favorable evaluation than other development tools for multiplatform application, like PhoneGap.

The evaluations are not totally accurate, though. The reason is that they are affected by so many variable factors that depend on the coding. The huge difference obtained can be helpful as a proof to demonstrate that the use of Bluetooth communication action XML tags favors easier and faster development processes for those applications that are mainly focused at Bluetooth communication.

But there are other very important factors in mobile applications besides the implementation complexity. The application’s performance is a key factor, as it determines the execution speed, especially in phones with limited processing capacity, and also...
affects battery consumption. The proposed solution simplifies the development of applications using a higher-level XML language. This approach involves a greater computational cost compared to using low-level languages, which can directly invoke the operating system API. To ensure the viability of the solution we must verify that the performance of the developed applications is acceptable for a Smartphone.

To analyze the performance we monitored several of the most relevant factors: memory and CPU usage. The Smartphone used to run the applications was a Galaxy Nexus. The device has the following technical features: OS – Android 4.2.2, Bluetooth v3.0 with A2DP, CPU – Dual-core 1.2 GHz Cortex-A9, Memory – 1 GB RAM. The tools used to monitor the performance parameters during the use of the applications were Advance Task Manager and SystemPanel.

First, we compared the performance of the web application that includes communication actions XML tags with two applications that offer the same Bluetooth communication capabilities: (1) Android – Java Android (a native application) and the hybrid application JavaScript – PhoneGap. The performance analysis was conducted during a reproduction of a use case; in this use case the applications established a connection with an electronic device and they performed ten information exchanges. The mean values of the results obtained in the first part of the performance analysis are shown in the Fig. 13. The CPU time spent and the memory consumption refer to analyzed applications and do not include the “Bluetooth Share”. The OS use a service called “Bluetooth Share”, which is responsible for execute the low level instructions when an application invokes the platform Bluetooth API. Bluetooth Share service assumes part of the computational cost of the communication (Fig. 12). This is not relevant to the analysis because in the three applications analyzed the service had identical computational cost, and this is because low-level Bluetooth communication was the same.

The result obtained in this part of the analysis is within expectations. The native application proved to be the most efficient, consuming 39.78% less memory than the web application with communication action XML tags. The hybrid applications developed with PhoneGap has also reduced the consumption of resources in 23.08% in comparison with the proposed solution. These performance differences are mainly due to two factors: (1) the Communicator web browser requires additional computational load to process the specific XML tags and execute the corresponding native API calls. Unlike the other two applications, the Communicator web browser has to do several actions (as loading the URL, downloading the HTML code, etc.) that are not related to the current functionality of the application, but also require computation. (2) The management and the graphic representation of user interfaces in HTML requires a higher computing capacity cost than native interfaces.

Secondly, we have compared the communicator web browser performance with several popular commercial web browsers. This analysis seeks to determine if the performance cost of the solution is acceptable for a Smartphone. The discussed use case has focused on the use of different browsers to access a web site that contains HTML code and seven communication action XML tags, although the Communicator Web browser was the only browser capable of processing such tags. The analysis’ results are presented in Fig. 14.
The Communicator Web browser has a significantly lower CPU and memory consumption than most of the analyzed web browsers. However, the result of this analysis should be interpreted with caution as they do not really know how many services or extra functionality are used by each commercial browser, features that may reduce the overall performance of the web browser. With the results of this performance analysis we can prove that the computational cost of the innovative functionality included in the Communicator Web browser is perfectly acceptable for a Smartphone, since the most popular web browsers have features and services with much higher computational costs.

6. Conclusions

In this paper we present a novel platform composed of a web browser and a specific markup language that could be considered as a framework. This specific markup language can be combined with common web technologies to develop web applications that will be able to use the Bluetooth communication module of the client device to exchange information in real time with nearby electronic devices, such as sensors, smart things, and personal computers.

Previously, the platforms that allow mobile data exchange with nearby electronic devices were native platforms (Android, Xcode, etc.) and, to a lesser extent, some hybrid platforms like PhoneGap. None of them was a web platform. This proposal offers the possibility of developing web applications that fuse the internet information with the information coming from nearby electronic devices. This new approach on the web opens the possibility of developing web applications that fuse the internet information with the information coming from nearby electronic devices. The proposal incorporates a relative simple language that simplifies the communication process; each tag is equivalent to many lines of code and each mobile platform is using its own API. As it has been evidenced in the analysis of the proposal, this goal has been successfully completed. By analyzing the proposal from a performance point of view, the proposed solution has higher memory consumption than native and hybrid applications, but compared to the most popular mobile web browsers, the computational cost derived from the innovative communication features functionality is not very high, and it is perfectly acceptable for a Smartphone.

The proposed markup language supports many common actions that allow the exchange of information in a restrictive manner and with a limited configurability.

7. Future work

Currently, some mobile devices are equipped with Radio-frequency identification (RFID) and Near field communication (NFC), these systems are very useful to obtain information from tagged physical objects. Web applications do not have mechanisms which take advantage of these technologies. We will investigate the best way to include RFID and NFC technologies in web-based systems.

References